Modeling and Control of a Humanoid Robot Using a Distributed Architecture Approach

Vítor M. F. Santos

University of Aveiro, Dept. of Mechanical Engineering, 3810-193 Aveiro, Portugal

Filipe M. T. Silva[1]

University of Aveiro, Dept. of Electronics and Telecommunications, 3810-193 Aveiro, Portugal

Abstract: ***This paper describes the methods and strategies to develop a humanoid robot using a distributed architecture approach where centralized and local control co-exist and concur to provide robust full monitoring and efficient control on a highly complex system with 22 DOF. A description of the hardware is given before the architecture is introduced because that influences greatly the methods implemented for the control and helps understanding the general decisions. The platform is still being developed, but the results are very promising, mainly because many approaches and research issues suddenly opened and will provide opportunities to test distributed control systems with possibilities that go far beyond the classical control of robots. The last part of the paper shows an example that is being developed to demonstrate the possibility of keeping a humanoid robot in upright balance position only by local control after reaction forces on the ground.***

Keywords: *Humanoid robots; Biped locomotion; Modular architectures; Distributed control.*

[1] Corresponding author's address: University of Aveiro, Department of Electronics and Telecommunications, Campus Universitário de Santiago, 3810-193 Aveiro, PORTUGAL (fsilva@det.ua.pt)

1. Introduction

The field of humanoid robotics has been attracting the attention of a growing community, both from the industry and academia. On the one hand, several companies have unveiled walking robots with impressive designs and skills, as the well-known Honda's ASIMO (Sakagami, Y., *et al.*, 2002) and Sony's QRIO (Nagasaka, K., *et al.*, 2004). On the other hand, the continuous progress in robotics technology opens up new possibilities for academic research on low-cost and easy-to-design humanoids, such as PINO (Yamasaki, F., *et al.*, 2000), ESYS (Furuta, T., *et al.*, 2001) and HanSaRam (Kim, J.-H., *et al.*, 2004), and others. Some platforms, however, show up limitations due to centralized control approaches and lack of modularity, making them difficult for others to replicate.

The main scope of the project beneath this paper has been the development of a humanoid platform to carry out research on control, navigation and perception. In particular, this paper focuses on the design and implementation of a distributed architecture for a humanoid robot where centralized and local controls co-exist and concur to provide a robust and versatile operation. The paper begins by presenting the design concepts and the technological solutions to build a small-size humanoid robot at reduced costs using off-the-shelf technologies, but still aiming at a fully autonomous platform for research. Afterwards, the software development and the integration techniques in building the proposed control architecture are described.

One most relevant feature of this implementation is the distributed architecture, supported on a CAN bus, in which independent and self-contained control units may allow either a cooperative or a standalone operation. The integration in these simpler control units of sensing, processing and acting capabilities play a key role to allow for localized control based on feedback from several sensors, ranging from joint position monitoring to force sensors. Moreover, the reprogrammable modules conduce to the central question of a true autonomy, i.e., the ability of self-control in which the robot may evolve over time. At the same time, the main advantage of the modular system is the possibility of reusing specific modules, in terms of both hardware and software, with no major efforts. Comparing with other architectures, even other based on CAN bus (Cho, Y.-J., *et al.*, 1999), stands out the high versatility of implementation and the easy expansibility at the system's level and reusable hardware, as described further.

2. The Humanoid Platform

2.1. Mechanical design

The platform has 22 degrees of freedom with 12 of them dedicated to the legs, which represent the most challenging part both for designing and control. Structure is made essentially of aluminum and steel for axles and other small components, weights about 6 kg and is about 60 cm tall. Fig. 1 shows a CAD model and a current stage of development.

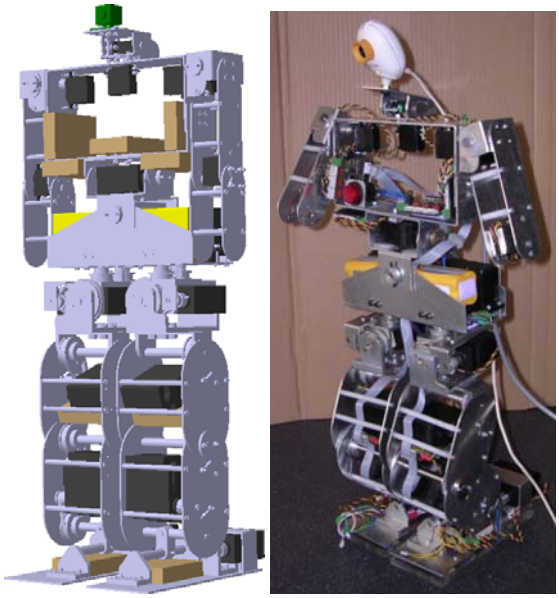


Fig. 1. 3D model of the humanoid robot and current stage of implementation

2.2. Actuators

Currently, the system actuators are 22 servomotors of three different types according to torque requirements of the several joints: more power on legs and less power on neck and arms. The option for servomotors is explained by the advantageous trade-off involving cost, power, dimensions and availability.

They are usually position controlled by a PWM at 50 Hz and duty-cycles around 1-2 ms but offer no velocity or torque control. That had to be done later with the several controllers in the architecture: velocity control is achieved by software using a dynamic PWM generation. That is, PWM is modified after feedback analysis of motor position until the necessary values are reached. The same can be done for torque control, but currently that is not yet implemented.

Since these servos, although the most powerful among their counterparts, offer torques not much higher than 2 Nm, gear transmissions had to be implemented in the mechanical structure.

2.3. Sensors and perception

Perception assumes a major role in an autonomous robot, therefore it must be reliable or abundant, or both if possible! For this platform the following perception was planned:

- Joint position (reading servo own potentiometer)
- Joint motor current (related to torque)
- Force sensors on the feet (ground reaction forces)
- Inclination of some links (using accelerometers)
- Angular velocity of some links (using a gyro)
- Vision unit (located on top)

Up to now, only vision has not yet been approached. The remainder sensors were addressed with different levels of accuracy, but all potentially usable with current hardware. Joint position is currently read

directly from the servomotor potentiometer. That signal, however, presents instability when the motor is moving which may affect measurement (some filtering is being done digitally). Possibly, in future developments an external potentiometer or encoder is to be used. Electric current consumption is easily measured as the voltage on the resistor (0.47 Ω) in series with the servo. Now follow some details on force and inertial sensors.

2.4. Foot force sensors

The foot sensors are intended to measure the force distribution on each foot to further assist during locomotion or simply keeping upright. Four sensors on each foot allow evaluating balance and a behaviour as the one illustrated in Fig. 2 is expected.

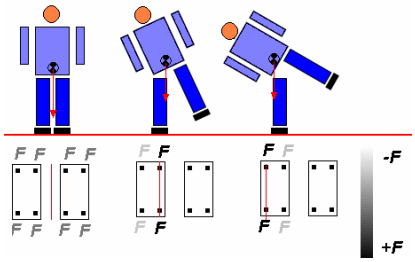


Fig. 2. Force sensors and balancing

Commercial force sensors are expensive, so it was decided to develop a system based on strain gauges and amplify the deformation of a stiff material. The result is a kind of foot whose details can be viewed in Fig.3 and is based on 4 acrylic beams located on the four corners of each foot that deform according to the robot posture. A simple Wheatstone bridge and an instrumentation amplifier complete the measuring setup (Fig. 4). The electronics hardware lays on a piggy-back board mounted on the local control unit, as can be seen in the lower part of Fig. 9.

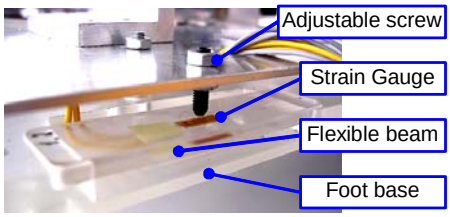


Fig. 3. Foot sensor details

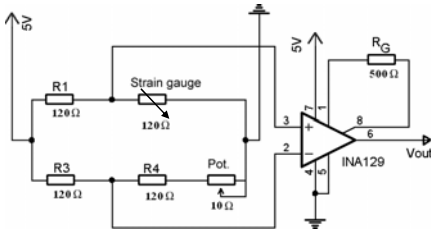


Fig. 4. Circuit to measure force on the feet

2.5. Inertial devices

Inertial perception is also a relevant source of information for dynamic and also static locomotion and balancing. Accelerometers and gyroscopes furnish information on acceleration and angular velocity.

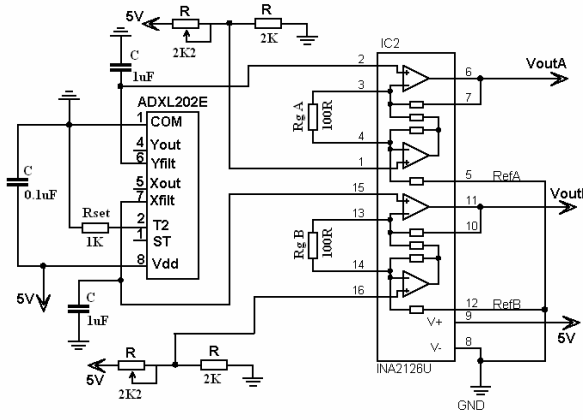


Fig. 5. Dual accelerometer electrical circuit

The accelerometers can be used to measure the acceleration of gravity, or better said, its component aligned with some axis. In other words, they can be used to measure inclination. That is what has been done by using the ADXL202E from Analog Devices. This very small MEMS device has two accelerometers in orthogonal axis that can be used to monitor tilt and roll angles of the platform. The system was mounted on a small PCB as shown in Fig. 6.

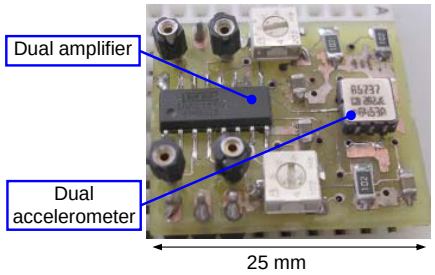


Fig. 6. Piggy-back board with two accelerometers

Finally, in what concerns sensors, a gyroscope unit has also been developed. The GYROSTAR ENJ03JA from MURATA has been selected due to several advantages and ease of interfacing. Up to now, the unit has not yet been used in the developed platform but its circuit is simply adapted from the vendor datasheet and using a INA129 amplifier.

3. Distributed architecture and software development

3.1. Principles of the approach

When the project begun, a main concern emerged: controlling a machine with so many degrees of freedom would be very demanding if done on a centralised control unit. Further, different levels of control would coexist making it difficult to develop new software and would also make debugging difficult. Last, but not the least, the endless web of wires and plugs connecting motors, sensors and central unit would make it not an easy task when assembling or reassembling components!

Besides these practical concerns, it made much more sense to enable the platform with simpler control units responsible for fewer tasks and therefore more robust to failure. All units would be interconnected by a local network so information could be exchanged among them in case it was necessary.

All this led to the conception of a distributed system. Without loss of generality of the approach, some motion joints have been grouped by vicinity criteria and are controlled locally by a dedicated board based on a PIC microcontroller. A CAN bus relays all units in a slave configuration.

Slave control units (SCU) generate PWM waves to control up to three actuators instead of only one. This rationalizes controllers in a practical implementation and also allows fast communication among three close joints by not depending on the bus. This represents an intermediate level of distributed system: not all actuators have fully independent hardware controllers. This concept was extended to perception. Once again, it would not be practical to have each of the several sensors with its own board hung up in the CAN bus. So, each slave unit has attached some nearby sensors that are somehow related to the actuators being controlled.

To collect data from the several SCUs, a special unit was conceived. Its role is to query all slaves for data and also give them directives or instructions for their behaviour. It was named the Master Control Unit since it accesses in first place the CAN bus and relays all SCUs. Its functions are simply to dispatch and collect data from the bus. Nonetheless, and being made from the same hardware as are the slaves, it could also control actuators and monitor sensors (those roles are refrained for now by a matter of principle!).

To end the architecture description a final element is necessary: the main or general control unit. In a first stage, this unit is the interface for the programmer with the remainder blocks and allows accessing data and issuing orders to actuators but, in the future, its role will be the general control directives, interface with remote systems and, very importantly, to process high rate data such as vision. It will also be able to change the SCU firmware and control programs, but progressively it is expected not to do so since the purpose is to proceed to a largely distributed control system, even with the possibility of self-learning (which, by the way, is possible with this hardware by using the Flash PIC controllers with on-board EEPROM). Fig. 7 illustrates a simplified representation of the architecture control units.

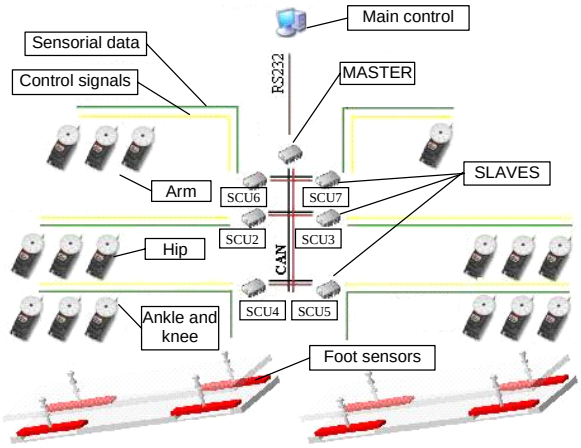


Fig. 7. Schematic of architecture layout

3.2. Hardware for the units in the architecture

As mentioned, master and slave units are based on a PIC microcontroller. Slaves are all alike and can be distinguished by a configurable address. Slaves can drive up to 3 servomotors, and can monitor their angular positions and electrical current consumption. Concerning additional sensors, each slave unit has the possibility of accepting a piggy-back board where additional circuit can lay to interface to other

sensors. Some examples of the developed piggy-back boards include force-sensors, accelerometers and gyroscope.

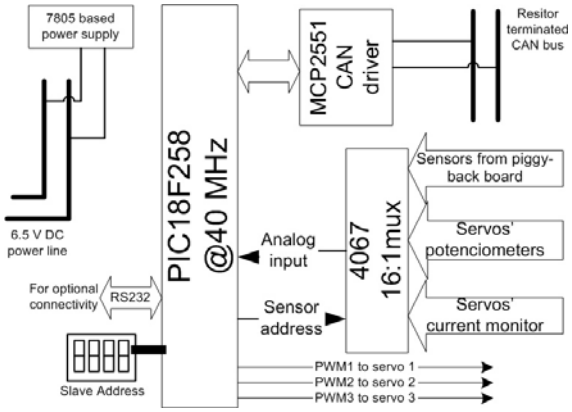


Fig. 8. Block diagram of generic slave unit

Fig. 8 shows a generic diagram of a slave unit. There, the main internal blocks can be seen, such as power supply regulation, CAN interface, the PIC controller, the multiplexer for sensor interfacing, PWM lines, CAN address switches and also lines prepared for RS232 communication. This kind of layout allows high versatility both on hardware and software approaches.

Being all similar, the construction of the boards is easier, along with software development (the same base code for all units). The master unit is different since it is not expected to drive motors neither to acquire many sensorial data. Furthermore, it communicates both by CAN and serial RS232 to the upstream controller. Hence, its piggy-back module was used to interface electrically the RS232 communications by installing a MAX232 circuit instead of sensor acquisition.

Slaves will be able to perform local control when adequate algorithms will be developed. In the slave units, three PWMs are generated for the three servomotors with resolutions of few micro-seconds according to directives received from the CAN bus, but local algorithms may decide better how to control the motors instead of relying on central control. Still at the slaves, the sensorial data is currently acquired with 8 bits, but 10 bits are possible in case it becomes necessary and adequate signal filtering and conditioning is provided. Fig. 9 shows a slave unit PCB with the main components and also includes a piggy-back board for signal electric conditioning. The RS232 plug is only used in the master to communicate with main control unit; slaves do not use it for now.

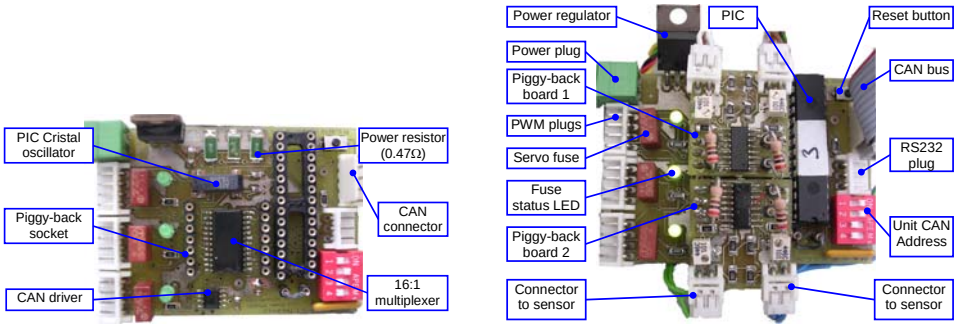


Fig. 9. The slave processing board without most components (right) and in full mounting with two piggy-back boards for force sensors (left).

3.3. Communications: CAN and RS232 messages

In the current stage of development, on power-up or reset, each slave checks its address and starts monitoring the CAN bus. While no messages arrive, the slave unit will drive its joints to a home position at a reduced speed and starts monitoring local sensors at a given rate (pre-programmed). When messages from the CAN bus arrive, the slave unit will process them; messages are of two kinds: imposing new desired position and speeds to each of the three motors, or query for sensorial data. These requests come from the master at a rate of 10 kHz throughout the entire cycle of slave units (currently, 8 units are used); the CAN bus is driven at 1 Mbit/s.

CAN messages contain a data field with 8 bytes, enough to exchange (in one message only) orders for three servos (3 positions and 3 speeds). On the other hand, to gather data from the slaves, more than one message may be necessary. Indeed from 3 motors 6 variables are required (3 positions and 3 current levels), and additional sensor values (such as force or inertial) must go on other messages. The master keeps a current status of the full system and delivers that data to the main control unit, when requested, using the RS232 link. Currently, this protocol defines a 4-byte message to the master and a 6-byte message from master to control unit.

One of the strengths of this architecture is that the main base code is the same for all slaves. Different behaviors are now decided by each SCU own address (dip-switch adjustable). Base code consists of generating PWM for motors obeying position and velocity requirements. Torque control (or something similar to that) appears possible since instant motor current can be monitored. Next developments will enable slaves to accept control directives and perform their own control accordingly.

4. Example of local control approach

4.1. General design approach

The open challenge is now to allow local controllers to perform actuator control based on sensor feedback and possibly a general directive. For instance, supposing that the global order is to keep balance in an upright position, although all actuators can intervene, the ankle and knee joints have a relevant role to keep an adequate force balance on each foot.

The relevant aspect of the proposed approach is the consideration of the ground reaction forces as control inputs for regulating the desired motion of the upper body. In this case, it is necessary that the controller may establish a relation between the desired mobility and the postural stability. Roughly speaking, it is up to the degrees of freedom nearest to the ground – ankle and knee – to assure the mobility and stability of the system, and to the degrees of freedom more distant from the ground – hip and trunk – the main role of compensation.

Here, we emphasis the role of a local controller, grouping the entire foot and knee joints, aiming to conciliate two imperatives: mobility and stability. The next subsections describe the control strategies applied to a simplified model used in simulation.

4.2. A simple model

The robot model attempted is a kinematics chain consisting of a planar two-DOF leg in contact with the ground. In the present study, we consider a simplified model that comprises a maximum height of

$L = 33cm$ and a total mass of $M = 5kg$ (see Table 1). Thus, the upper body and the swing leg masses are, in some way, embedded into this simplified model.

Robot Link	Mass (*kg*)	Length		Spring-damper model	
		l_i (*m*)	r_i (*m*)	K_x (*N/m*)	B_x (*Ns/m²*)
Thigh	4.0	0.165	0.09	50.0×10^3	1000.0
Shank	0.6	0.165	0.12	K_y (*N/m*)	B_y (*Ns/m²*)
Foot	0.4	0.12	0.04	200.0×10^3	250.0

Table 1 – Robot and environment parameters.

It is assumed the existence of two actuators (ankle and knee) and two contact points where the force sensors are inserted (Fig. 10). The contact of the foot with the constraint surface is modelled through a linear spring-damper system in the horizontal and vertical directions. Further, the friction is assumed to be large enough to avoid any foot's slippage.

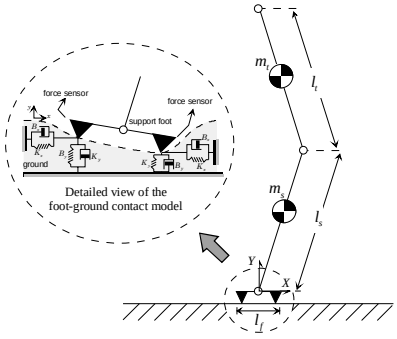


Fig. 10. Planar 2-DOF foot-leg model and constraint surface.

4.3. Force interaction control

The proposed control algorithm is based on simple motion goals taking into account the reaction forces between the feet and the ground. The two variables to be controlled are the normal reaction forces across the heel and at the toe, f_n^{heel} and f_n^{toe}, respectively. The reference signals are generated automatically in result of demands (motion goals) imposed to the upper body section. In other words, the proposed control combines both force feedback with online pattern-modifications.

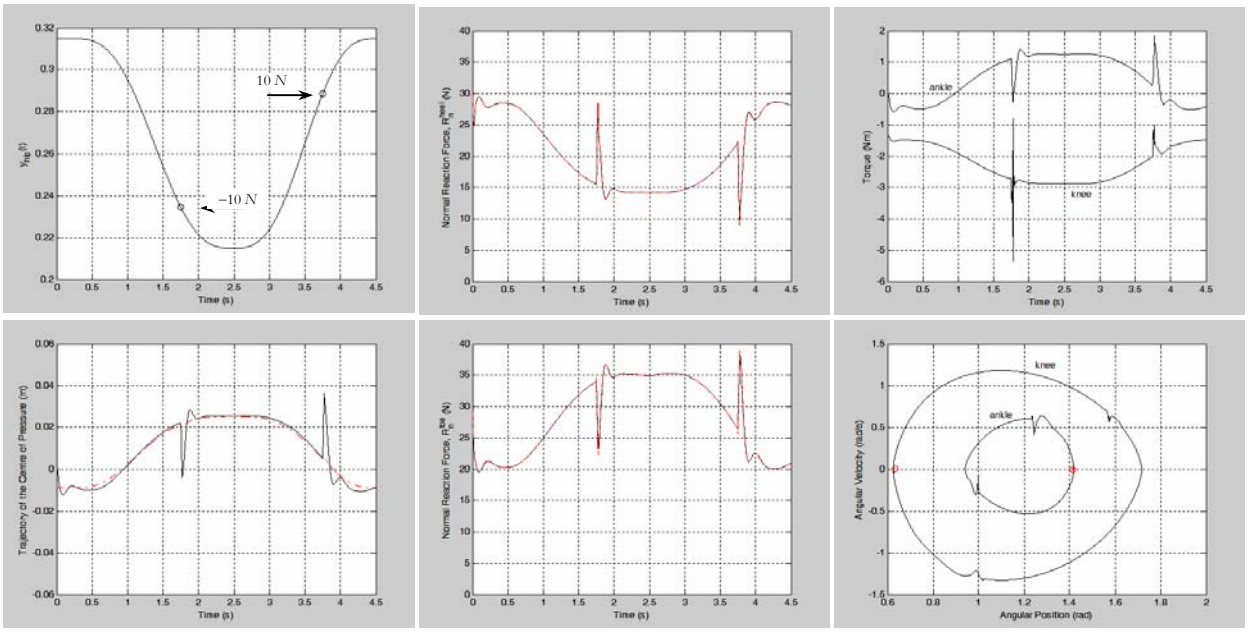
The desired normal force is calculated from the errors in the vertical hip pose using a linear controller:

$$f_n^d = BW + \left[K_p^f \left(y_{hip}^d - y_{hip} \right) + K_v^f \left(\dot{y}_{hip}^d - \dot{y}_{hip} \right) \right] \tag{1}$$

where *BW* is the total system's weight, K_p^f and K_v^f are appropriate constant feedback gains. On the other hand, the desired location of the centre of pressure is calculated as function of the horizontal hip pose as:

$$COP^d = K_p^{COP} \left(x_{hip}^d - x_{hip} \right) + K_v^{COP} \left(\dot{x}_{hip}^d - \dot{x}_{hip} \right) \tag{2}$$

This means that the reference COP is actively used to calculate the distribution of the total reaction force along the extremities of the support foot. These are the variables that some force control algorithm must follow. In this study, it is used a fractional-order controller combined with a genetic algorithm for optimal tuning of the control parameters. A more detailed explanation of the controller can be found elsewhere (Silva, F. & Santos, V., 2005).



(*a*) (*b*) (*c*)

Fig. 11. Temporal evolution of the: (*a*) desired vertical hip motion (up) and centre of pressure (down); (*b*) real *vs.* desired normal ground reaction forces; (*c*) joint torques (up). Phase plane: (*c*) ankle and knee joints (down).

4.4. Simulation results

In order to verify the effectiveness of the local controller, several simulations are carried out addressing the problem of standing posture in face of external perturbations. In this case, the motion planning is accomplished by prescribing the Cartesian trajectories of the hip. The double inverted pendulum is standing, moves down vertically and up again to the initial posture. The initial position, in meters, is set to $\left(x_{hip}, y_{hip}\right) = \left(-0.05, 0.31\right)$ and the desired *x*-coordinate remains constant along the motion. The performance of the controller is evaluated by applying two external perturbations. Here, the perturbation corresponds to a horizontal force of ±10 *N* applied at the hip during 20*ms* (see Fig. 11-*a*).

From the results, a few remarks ought to be made. First, the force controller is effective to generate the desired hip motion, while the COP remains inside the support covered by the stance foot (Fig. 11-*a*). Second, the temporal evolution of the normal reaction forces reflects the COM accelerations (Fig. 11-*b*). Third, the system state converges to the attractor at the neighborhood of the zero angular velocity in the phase plane (Fig. 11-*c*).

5. Conclusion and perspectives

This paper described the development and integration of software components to build a humanoid robot based on off-the-shelf technologies. The design considerations that governed this project are based on modular and reusable principles. The main features are the distributed control architecture and the relevance given to the reaction forces information in order to achieve a better adaptive behaviour.

Ongoing developments cover the inclusion of vision and its processing, possibly with a system based on PC104 or similar. The future research will cover distributed control, alternative control laws and also deal with issues related to navigation of humanoids and, hopefully, cooperation. In this sense, it seems

essential to combine the force control techniques with more advanced algorithms such as adaptive and learning strategies.

6. Acknowledgments

The authors would like to thank the following undergraduate students in Mechanical Engineering at the University of Aveiro for their contributions in the humanoid hardware and software implementation: David Gameiro, Filipe Carvalho, Luis Rego, Renato Barbosa, Mauro Silva, Luis Gomes, Ângelo Cardoso and Nuno Pereira.

7. References

Cho, Y.-J., *et al.*, 1999, "A Compact/Open Network-Based Controller Incorporating Modular Software Architecture for a Humanoid Robot", Journal of Intelligent and Robotic Systems, 25: 341–355.

Furuta, T., *et al.*, 2001, "Design and Construction of a Series of Compact Humanoid Robots and Development of Biped Walk Control Strategies", Robotics & Autonomous Systems, Vol. 37, pp. 81-100.

Kim, J-H., *et al.*, 2004, "Humanoid Robot HanSaRam: Recent Progress and Developments", Journal of Computational Intelligence, Vol. 8, nº 1, pp. 45-55.

Nagasaka, K., *et al.*, 2004, "Integrated Motion Control for Walking, Jumping and Running on a Small Bipedal Entertainment Robot", in Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3189-3194.

Sakagami, Y., *et al.*, 2002), "The Intelligent ASIMO: System Overview and Integration", in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2478-2483.

Silva, F.M., and Santos, V.M., 2005, "Towards an Autonomous Small-Size Humanoid Robot: Design Issues and Control Strategies", in Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, Espoo, Finland.

Yamasaki, F., *et al.*, 2000, "PINO the Humanoid: a Basic Architecture", in Proceedings of the International Workshop on RoboCup, Australia.

List of Figures